Bullock, S., Cartlidge, J. P., & Thompson, M. (2002). Prospects for Computational Steering of Evolutionary Computation. In E. Bilotta, G. D., T. Smith, T. Lenaerts, S. Bullock, H. H. Lund, J. Bird, R. Watson, P. Pantano, L. Pagliarini, H. Abbass, R. Standish, & M. Bedau (Eds.), *Artificial Life VIII: Workshop Proceedings of the Eighth International Conference on Artificial Life: Beyond Fitness - Visualising Evolution* (pp. 131-137). Massachusetts Institute of Technology (MIT) Press.

Peer reviewed version

Prospects for Computational Steering of Evolutionary Computation

Seth Bullock, John Cartlidge and Martin Thompson

Informatics Research Institute, School of Computing, University of Leeds, Leeds, LS2 9JT, UK

seth@comp.leeds.ac.uk, johnc@comp.leeds.ac.uk, mart@comp.leeds.ac.uk

Abstract

Currently, evolutionary computation (EC) typically takes place in batch mode: algorithms are run autonomously, with the user providing little or no intervention or guidance. Although it is rarely possible to specify in advance, on the basis of EC theory, the optimal evolutionary algorithm for a particular problem, it seems likely that experienced EC practitioners possess considerable tacit knowledge of how evolutionary algorithms work. In situations such as this, computational steering (ongoing, informed user intervention in the execution of an otherwise autonomous computational process) has been profitably exploited to improve performance and generate insights into computational processes. In this short paper, prospects for the computational steering of evolutionary computation are assessed, and a prototype example of computational steering applied to a coevolutionary algorithm is presented.

Introduction

The field of evolutionary computation (EC) is principally concerned with designing powerful search algorithms modelled on natural evolution. Such algorithms feature a population of individuals subjected to processes analogous to the competition, selection, reproduction, mutation, etc., that drive natural evolution. When successful, these processes gradually alter the population until it comprises high-quality solutions to the problem that the individuals are competing to solve.

EC is used for combinatorial, numerical optimization and design problems such as scheduling, routing, data mining, software agent design, robot control, and optimization for engineering design in general (e.g., Gen & Cheng, 1997). It is particularly useful when problems are poorly understood and large enough to render exhaustive search or analytic approaches intractable.

The evolving populations involved in EC are high-dimensional, time-varying systems that exhibit complex and often counter-intuitive dynamics across a variety of time-scales and at many different levels of organisation. Understanding how and why such systems behave in the ways that they do is crucial if we are to harness the creative and design potential of evolutionary algorithms. While there has been considerable effort towards developing EC theory capable of explaining how, why, and when EC techniques are effective, we are still some way from possession of anything more useful than guidelines and rules of thumb.

Discovering efficient and intuitive ways to visualize the behaviour of these systems is one method of generating insights into the way that they work. Surprisingly, there has been relatively little well-founded research in this area. While members of the EC community have developed many idiosyncratic (and often short-lived) graphing techniques with which to display the behaviour of particular systems, overall there remains a reliance on rather simplistic plots of population summary statistics over time—visualizations that by their nature disguise much of the system complexity.

In this short paper, the state of EC visualization will first be briefly reviewed, before its potential role in facilitating *computational steering* is introduced and discussed. An example of a simple application of computational steering to a coevolutionary system will be presented. The prospects for successful application of computational steering techniques to evolutionary algorithms in general will be assessed, and the potential for this approach to progress EC theory and the exploitation of EC in industrial and commercial contexts will be estimated.

EC Visualization

The recent increase in affordable computing power (in terms of speed, storage, and graphics) has seen a growing interest in increasingly sophisticated EC visualization. For recent reviews, see Hart and Ross (2001) and Collins (in press).

The techniques that have been developed to visualize evolutionary algorithms can be categorised in several ways. Some are snapshots of algorithm performance over time presented postmortem as a single, static view, while some are dynamic animations depicting "on-line" the way in which the algorithm changes over time. Some represent the evolving population, some the character of the problem being solved. Visualizations have been

developed at several levels of description, from presenting population-level summary statistics, through distributions of individual-level variables, to gene- or loci-level representations.

While a wide variety of visualization tools have been developed across these categories, few have been published outside of technical reports (e.g., Kapsalis & Smith, 1992; Dabs & Schoof, 1995; Collins, 1998b; Bosman, 1999; Wu, De Jong, Burke, Ramsey, & Grefenstette, 1999; Pohlheim, 2001; Hart & Ross, 2001). Often these visualizations were built in order to understand the behaviour of a specific EC system as that system was developed, and were subsequently generalized to some degree. As a result, the intended user audience typically appears to be EC researchers studying how algorithms work, rather than industrialists exploiting EC to solve particular problems, modellers using EC to simulate natural systems, or EC tutors teaching evolutionary algorithms to students. Moreover, the needs and abilities of the intended users are rarely considered explicitly, and there has been little exploration of the usability or effectiveness of the visualization systems. In the few cases where these issues have been addressed, it has been in terms of requirements capture from EC users (Collins, 1998a), and usability assessment in an experimental setting (Wiles & Tonkes, 2002).

The range of visualization tools employed includes standard techniques such as various types of multidimensional scaling (e.g., Spears, 1994; Collins, 1999; Pohlheim, 1999), Sammon mapping (Sammon 1969; Dawinel 1994; employed by Dybowski, Collins, & Weller, 1996) and quadcodes (Li & Loewn, 1987; independently developed by Collins, 1997; Shine & Eick 1997; Wiles & Tonkes, 2002), and entirely novel approaches developed specifically for dealing with EC issues such as techniques for representing genotypic changes over evolutionary time (Wu et al., 1999; Hart & Ross, 2001). None of these platforms or techniques has achieved significant penetration in the EC community, as of yet.

A largely independent stream of research relies on EC theory to inform visualization by suggesting what type of data will be informative, in what way, and in which situations. In addition to generating useful visualization tools, this approach is intended to progress EC theory by providing a richer appreciation of the behaviour of EC systems. When successful, this integration should result in *cross-fertilisation* of theory and visualization, with ideas, techniques and insights from each driving forward the other.

Studies that have combined EC visualization and theory in this way are still relatively rare. For instance, Cliff and Miller (1995) propose a visualization technique for detecting cycling in coevolutionary systems, Harvey and Thompson (1996) explore the use of various visualization methods in order to explain the role of neutral ridges in the evolutionary search space, Bedau and Brown (1998) visualize an evolutionary activity metric, and Bullock (2001) uses visualization to demonstrate the biases inherent in a range of mutation operators.

In summary there is thus considerable potential to build on these initial studies. Perhaps the most pressing issues concern requirements capture and formal evaluation. As yet the different needs and abilities of the various types of EC end-user (novices, experts, researchers, industrialists, tutors, students, etc.) are not well-characterised. Moreover, the usability and effectiveness of existing visualization techniques are poorly understood. More attention to both of these aspects of EC visualization research are necessary in order to develop tools that effectively meet user requirements.

One further under-explored area of research will be discussed below, that of developing *interactive* visualization tools to support computational steering.

EC Steering

EC algorithms are typically executed in "batch mode". A stereotypical scenario might be as follows: 1. parameters are set and decisions are made regarding the type and amount of data that the algorithm will produce as output. 2. the algorithm is then executed, often taking hours if not days or weeks of compute time. 3. during execution there is little if any interaction with the system—output may be graphed intermittently; the search process may be terminated if it appears to have failed. 4. upon completion the solutions produced by the algorithm are assessed, and any data output by the algorithm is graphed. 5. on the basis of this information, parameters may be altered, or more significant changes may be made to the algorithm, before it is executed again, hopefully with improved performance.

A combination of batch processing and simple graphs of population summary statistics changing over evolutionary time is inadequate in two respects: (i) significant interaction can only take place once the algorithm has (been) terminated, (ii) the information upon which this interaction is based is crude and fragmented.

At the opposite extreme, some interactive evolutionary algorithms require user guidance at every generation in the form of artificial selection—the user must choose which members of the population are good enough to contribute genetic material to the next generation. Without user instruction, such algorithms halt. Understandably, given the demands on user time imposed by algorithms of this kind, they are typically only considered in cases where it is hard or impossible to operationalise an accurate fitness function, e.g., where fitness is subjective. Examples include the aesthetic evolution of a solution tailored to a particular person's taste, or of works of art (Dorin, 2001).

Computational steering—defined here as the ongoing,

informed intervention of users in the execution of an otherwise autonomous computational process (see, e.g., Parker, Johnson, & Beazley, 1997)—lies somewhere between these two extremes. Just as air-traffic controllers engage in a continuous dialogue with the planes that they are responsible for, guiding their behaviour via a series of heavily stereotyped interactions on the basis of information relayed to them in a variety of ways, so computational steering applications allow users to manually interact with or "steer" computational processes. It is important to recognise that, just as planes continue to fly as best they can in the absence of air-traffic control, during computational steering a user does not *determine* the behaviour of a computational process, but rather only influences it. Typical interventions might include altering system parameters, turning on or off aspects of the algorithm, etc.

As with the manually-driven aesthetic evolution described above, computational steering tends to be used in situations where the skills and knowledge of human users are critical to system performance, but are difficult or impossible to operationalise as computer code. For example, the pattern recognition skills of air-traffic controllers are key to the success of air-traffic control systems. If it were possible to replicate these skills computationally we might not need to rely on humans in the system at all.

Prospects

Are there reasons to believe that enabling people to steer evolutionary algorithms is desirable, let alone attainable? Successful computational steering could achieve two important results. First, effective steering may improve system performance in terms of efficiency and quality. Second, computational steering may lead to insights into how the system works, making explicit the tacit knowledge that users employ in their steering behaviour. In the case of evolutionary computation, we are sorely in need of both. Although the evolutionary computation community remains healthy, there is little evidence that there are important classes of search or optimization problems for which it can persuasively be claimed that evolutionary algorithms are likely to outperform competing approaches. That this is the case is largely due to the lack of theoretical insights into what makes evolutionary algorithms work when they do. As such, computational steering offers a significant opportunity to progress evolutionary computing on two important and inter-related fronts.

In addition, there are some secondary reasons for pursuing this line of research. First, computational steering will require sophisticated visualization of evolutionary algorithms, a worthy research aim in its own right. Second, computational steering is likely to be of use in teaching novices how evolutionary algorithms behave (and perhaps how evolution itself works).

But what chance is there that it will work? In order for computational steering to succeed, users must be able to make effective interventions. Three conditions must be met in order to achieve this. First, users must possess knowledge, understanding, skills, etc. that could be used to effectively steer computation (expertise). Second, users must be able to exert the required influence on the system in an intuitive manner (interaction). Third, they must be presented with the information that they require in order to make these interventions in a similarly intuitive manner (visualization).

It is as yet unclear to what extent the existing EC visualization techniques described above meet this last condition. What information might EC users require in order to make effective interventions? Measures of population make-up such as diversity (Bedau, Ronneburg, & Zwick, 1992), evolutionary activity (Bedau & Brown, 1998), etc? Measures of landscape properties such as ruggedness (Weinberger, 1990; Stadler, 1996; Hordijk, 1997; Vassilev, Fogarty, & Miller, 2000), neutrality (Barnett 1998; Bullock 2001, in press; Smith, Husbands, Layzell & O'Shea, 2002), evolvability (Smith, Husbands, & O'Shea, 2001), etc? Some success has been achieved applying computational steering to design optimization algorithms by using quality-coloured representations of trajectories in design space (Wright, Brodlie, & David, 2000), but much remains to be done in this area.

It is perhaps useful to divide the work that has been devoted to developing rich modes of interaction with evolutionary algorithms, of which there has been relatively little, into two aspects. First, users need to be able to manipulate the visualizations that they are provided with. Amongst others, Collins (1998a) has introduced the basic elements of this type of interaction, e.g., alpha-sliders with which to dynamically alter the portion of the data set being viewed, and the various scales at which it is represented. The second aspect of EC interaction is largely unexplored: real-time alteration of an EC algorithm's operation through, for instance, changing parameter values (e.g., mutation rate, population size), turning on/off aspects of an algorithm (e.g., sexual reproduction, elitism), or exerting temporal control through rewinding, restarting, etc. Although this type of research will mostly involve human-computer interaction issues, there are also practical concerns. How much user time would need to be devoted to steering an evolutionary algorithm? At what points in an evolutionary run is steering most profitable? How much expertise is required in order to effectively steer an evolutionary algorithm? Given the highly parallel nature of the processes being steered, might collaborative steering be appropriate (Wood, Wright, & Brodlie, 1997)?

However, it is the first of the three conditions listed above that has received the least consideration (at least

in print). Just how likely is it that expert EC users have the wherewithal to successfully guide evolutionary processes? Is there a wealth of tacit knowledge that EC experts could successfully bring to bear on EC algorithms as they execute? Given the gap that exists between the published canon of formal EC theory and the working knowledge possessed by the EC community, there are grounds to believe that the answers are positive. However, as yet we simply do not know.

An Example

As an illustrative example, we will briefly present a very simple application of computational steering to a coevolutionary problem. The software is currently in the early stages of development, but will serve out purposes here.

Previous research has explored the problem of coevolutionary disengagement: periods during which one coevolving population outperforms its coevolutionary opponent population to the extent that competing individuals achieve near-identical fitness scores (Watson & Pollack, 2001). During these periods, selection pressure disappears and populations suffer from evolutionary drift, stagnating as deleterious mutations are accumulated. Avoiding coevolutionary disengagement (along with over-focusing and cycling) could significantly improve the performance of coevolutionary optimization algorithms.

It has been demonstrated that reducing population "virulence" (the extent to which members of one population are selected to maximally defeat members of an opponent population) lowers the probability of disengagement (Cartlidge & Bullock, 2002). When populations are highly engaged, strong antagonistic selection pressures of the kind normally implemented in competitive coevolutionary search algorithms drive populations forward. However, as coevolving populations become less engaged, selection for increasingly mild virulence (i.e., selecting for individuals who achieve less than 100% success against their coevolutionary opponents) encourages increased engagement and thereby reduces the chance of the stagnation that accompanies evolutionary drift. These results, although perhaps counterintuitive in the context of evolutionary optimization, are reminiscent of natural parasite-host coevolution, where it is often not in the best interests of a parasite to maximally exploit its host.

As yet, we do not understand how to automatically vary coevolutionary selection pressures such that they suit the degree of engagement that populations experience. In order to move towards such an automatic algorithm, we are pursuing a computational steering approach in which the selection pressures exerted on hosts and parasites as they coevolve are under the control of a user. On the basis of visually presented information concerning the fitness distributions being achieved by each population, users must vary two algorithm parameters in an effort to maintain engagement and maximise population fitness.

Figure 1 depicts the steering interface. The two plots at the right of the panel present graphs of fitness over time, and are updated as the populations coevolve. The two plots at the top left of the panel depict the character of the selection pressure exerted on each coevolving population. Sliders beneath these latter plots allow the user to vary selection pressure. The information in the lower left portion of the panel concern various algorithm parameters.

This platform will be used to carry out several kinds of study. For some purposes, parts of the interface can be removed, or fixed. In this way, more of less complex interfaces can easily be constructed. First, experiments will compare the performance of naive users with experts as they attempt to effectively steer coevolutionary optimization under various conditions. These subjects will be carefully debriefed in an effort to understand how they went about their task. Any insights gained could be used to design more efficient automatic coevolutionary algorithms.

In addition, the platform will be used directly by researchers as a tool with which to explore coevolutionary disengagement. The nature of the platform allows a rich interaction with the coevolutionary system, which will hopefully lead to improved insights into algorithm behaviour. Finally, the steering platform will be used to present results in an intuitive way at meetings, etc., and to teach coevolutionary concepts to students in a hands-on manner that will hopefully be engaging and informative.

It must be noted that this example of computational steering is very crude in many respects. The visualizations used to inform the user of algorithm behaviour are for the most part standard plots of fitness over time. In general, much more sophisticated indicators of a wider range of system aspects could be employed. Similarly, the range of interaction supported by the example steering platform is limited to varying a small number of the parameters governing evolutionary selection pressures. There are, of course, many alternative aspects of algorithm performance that could be influenced by the user. However, we expect the steering interface to be an adequate tool for the specific purposes of exploring the role of virulence in coevolutionary engagement.

Conclusions

Working towards effective computational steering for evolutionary computation would appear to have considerable merit if there is some chance of achieving the improvements in algorithm performance and insights into algorithm behaviour that it may bring. However, even if such a research effort were to fail in its stated pri-

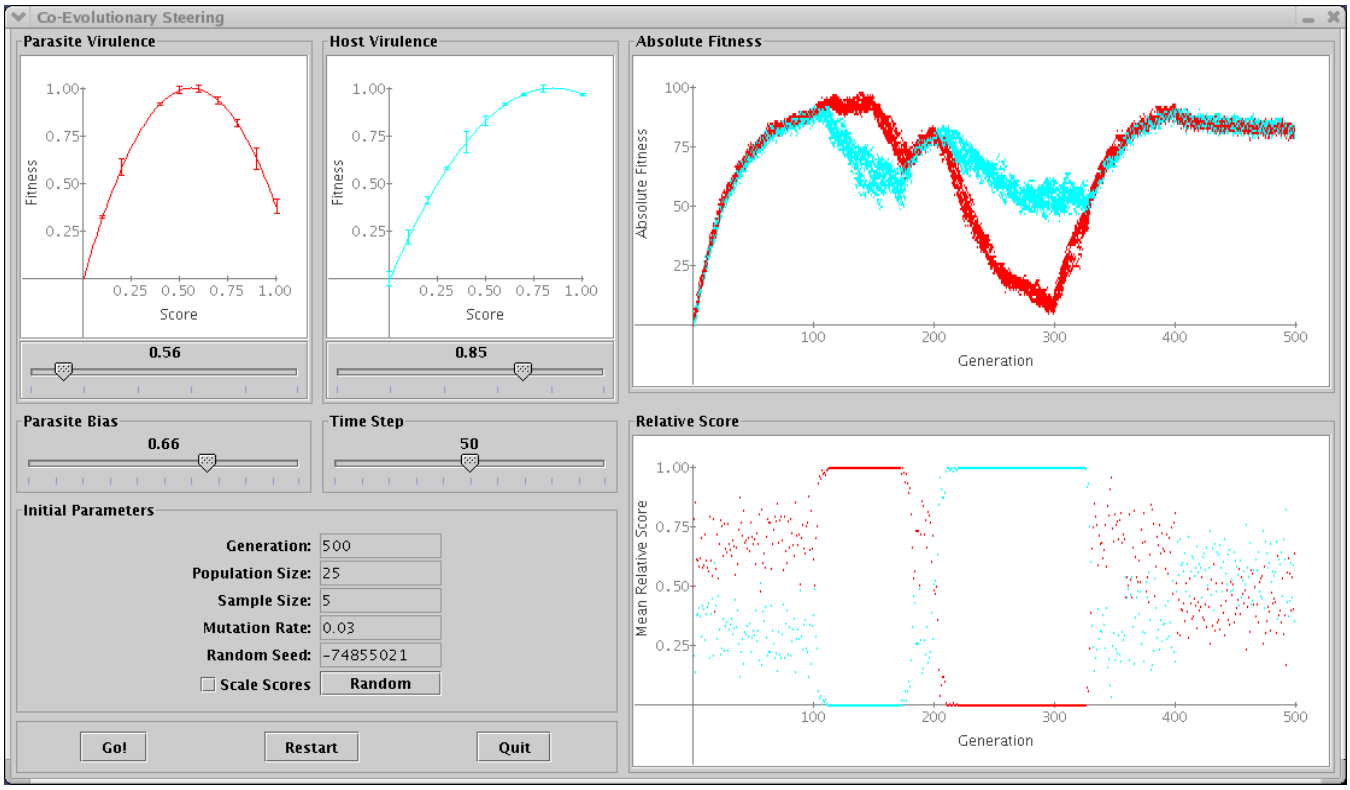

Figure 1: An interface for steering a coevolutionary algorithm. See text for details.

mary aim, in pursuing it several secondary goals would be progressed.

First, effective visualization and interaction are worthy targets in their own right, as the other papers at this workshop testify. Second, and perhaps more importantly, in integrating visualization, interaction and EC theory, research into computational steering for evolutionary computation has the potential to improve our understanding of all three. To the extent that visualization and interaction techniques are motivated by some theory of evolutionary computation, the ability of these techniques to support computational steering is a good indicator of the success of EC theory in guiding our understanding of evolutionary algorithms.

Acknowledgements

This paper benefitted from discussion with Ken Brodlie, Roy Ruddle, Tom Smith and conversation with the University of Leeds' biosytems reading group. However, any opinions stated here are our own.

References

Barnett, L. (1998). Ruggedness and neutrality—the NKp family of fitness landscapes. In Adami, C., Belew, R., Kitano, H., & Taylor, C. (Eds.), *Artificial Life VI*, pp. 18–27. MIT Press.

Bedau, M. A., Ronneburg, F., & Zwick, M. (1992). Dynamics of diversity in an evolving population. In Maenner, R., & Manderick, B. (Eds.), *Parallel Problem Solving from Nature*, Vol. 2, pp. 94–104. Elsevier.

Bedau, M. A., & Brown, C. (1998). Visualizing evolutionary activity of genotypes. Technical report 98-03-23, Santa Fe Institute, Santa Fe, NM.

Bosman, P. A. N. (1999). EA visualizer tutorial. Technical report UU-CS-1999-20, Utrecht University.

Bullock, S. (2001). Smooth operator? Understanding and visualising mutation bias. In Kelemen, J., & Sosik, P. (Eds.), *Sixth European Conference on Artificial Life*, pp. 602–612. Springer Verlag.

Bullock, S. (in press). Will selection for mutational robustness significantly retard evolutionary innovation on neutral networks?. In *Artificial Life VIII*. MIT Press.

Cartlidge, J., & Bullock, S. (2002). Learning lessons from the common cold: How reducing parasite virulence improves coevolutionary optimization. In Fogel, D. (Ed.), *Congress on Evolutionary Computation*, pp. 1420–1425. IEEE Press.

Cliff, D., & Miller, G. F. (1995). Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In Morán, F., Moreno, A., Merelo, J. J., & Chacón, P. (Eds.), *Third European Conference on Artificial Life*, pp. 200–218. Springer.

Collins, J. J. (1999). Visualization of evolutionary algorithms using principal component analysis. In Wu, A. S. (Ed.), *Evolutionary Computation Visualization Workshop – Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program*. Morgan Kaufmann.

Collins, T. D. (1997). Using software visualization technology to help evolutionary algorithm users validate their solutions. In Bäck, T. (Ed.), *Seventh International Conference on Genetic Algorithms*, pp. 307–314. Morgan Kaufmann.

Collins, T. D. (1998a). *The application of software visualization technology to evolutionary computation*. Ph.D. thesis, Knowledge Media Institute, The Open University, Milton Keynes, UK.

Collins, T. D. (1998b). Understanding evolutionary computing: A hands on approach. In *IEEE International Conference on Evolutionary Computation*, pp. 564–569. IEEE Press.

Collins, T. D. (in press). Visualizing evolutionary computation. In Ghosh, A., & Tsutsui, S. (Eds.), *Advances in Evolutionary Computation*. Springer Verlag.

Dabs, T., & Schoof, J. (1995). A graphical user interface for genetic algorithms. Technical report 98, Lehrstuhl fur Informatik II, University of Wurzburg, Germany.

Dorin, A. (2001). Aesthetic fitness and artificial evolution for the selection of imagery from the mythical infinite library. In Kelemen, J., & Sosik, P. (Eds.), *Sixth European Conference on Artificial Life*, pp. 659–668. Springer.

Dybowski, R., Collins, T. D., & Weller, P. (1996). Visualization of binary string convergence by sammon mapping. In Fogel, L., Angeline, P., & Bäck, T. (Eds.), *Fifth Annual Conference on Evolutionary Programming*, pp. 377–383. MIT Press.

Dzwinel, W. (1994). How to make Sammon's mapping useful for multidimensional data structure analysis. *Pattern Recognition*, *27*, 949–959.

Gen, M., & Cheng, R. (1997). *Genetic Algorithms and Engineering Optimization*. John Wiley & Sons.

Hart, E., & Ross, P. (2001). GAVEL: A new tool for genetic algorithm visualization. *IEEE Transactions on Evolutionary Computation*, *5*(4), 335–348.

Harvey, I., & Thompson, A. (1996). Through the labyrinth evolution finds a way: A silicon ridge. In Higuchi, T., Iwata, M., & Weixin, L. (Eds.), *First International Conference on Evolvable Systems*. Springer-Verlag.

Hordijk, W. (1997). A measure of landscapes. *Evolutionary Computation*, *4*(4), 335–360.

Kapsalis, A., & Smith, G. (1992). *The GAmeter Development Toolkit User Interface Manual*. University of East Anglia, Norwich, UK.

Li, S., & Loew, M. H. (1987). The quadcode and its arithmetic. *Communications of the ACM*, *30*(7), 621–626.

Parker, S. G., Johnson, C. R., & Beazley, D. (1997). Computational steering software systems and strategies. *IEEE Computational Science & Engineering*, *4*(4), 50–59.

Pohlheim, H. (1994-2001). GEATbx: Genetic and evolutionary algorithm toolbox for matlab. http://www.geatbx.com/.

Pohlheim, H. (1999). Visualization of evolutionary algorithms: Set of standard techniques and multidimensional visualization. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Genetic and Evolutionary Computation Conference*, pp. 533–540. Morgan Kaufmann.

Sammon, J. (1969). A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, *C-18*(5), 401–408.

Shine, W., & Eick, C. (1997). Visualizing the evolution of genetic algorithm search processes. In *IEEE International Conference on Evolutionary Computation*, pp. 367–372. IEEE Press.

Smith, T. M. C., Husbands, P., Layzell, P., & O'Shea, M. (2002). Fitness landscapes and evolvability. *Evolutionary Computation*, *10*(1), 1–34.

Smith, T. M. C., Husbands, P., & O'Shea, M. (2001). *Not* measuring evolvability: Initial investigation of an evolutionary robotics search space. In *Congress on Evolutionary Computation*, pp. 9–16. IEEE Press.

Spears, W. (1994). Visualizing genetic algorithms. Technical report AIC-94-055, AI Center, Naval Research Laboratory, Washington, DC.

Stadler, P. F. (1996). Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, *20*, 1–45.

Vassilev, V. K., Fogarty, T., & Miller, J. (2000). Information characteristics and the structure of landscapes. *Evolutionary Computation*, *8*(1), 31–60.

Watson, R., & Pollack, J. (2001). Coevolutionary dynamics in a minimal substrate. In Spector, L. (Ed.), *Genetic and Evolutionary Computation Conference*, pp. 702–709. Morgan Kaufmann.

Weinberger, E. (1990). Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, *63*, 325–336.

Wiles, J., & Tonkes, B. (2002). Visualisation of hierarchical cost surfaces for evolutionary computation. In Fogel, D. (Ed.), *Congress on Evolutionary Computation*, pp. 157–162. IEEE Press.

Wood, J., Wright, H., & Brodlie, K. (1997). Collaborative visualization. In Yagel, R., & Hagen, H. (Eds.), *IEEE Visualization*, pp. 253–260. ACM Press.

Wright, H., Brodlie, K., & David, T. (2000). Navigating high-dimensional spaces to support design steering. In Yagel, R., & Hagen, H. (Eds.), *IEEE Visualization*, pp. 291–296. ACM Press.

Wu, A. S., De Jong, K. A., Burke, D. S., Ramsey, C. L., & Grefenstette, J. J. (1999). VIS: A genetic algorithm visualization tool. In Wu, A. S. (Ed.), *Evolutionary Computation Visualization Workshop – Proceedings of the Genetic and Evolutionary Computation Conference Workshop Program*. Morgan Kaufmann.